*Application Note*

# *I²C 基本指南*

*Joseph Wu*

## 摘要

微控制器和不同外设之间的通信需要某种数字协议。I²C 是一种通用通信协议，用于 TI 生产的许多不同产品系列的各种器件。本应用手册首先对 I²C 协议进行了基本概述，介绍了协议的发展历程、不同的 I²C 速度模式、数字通信的物理层和数据结构。文中展示了不同数据转换器之间通信协议的几个示例。最后，本应用手册涵盖了协议的一些不常见方面，包括保留地址、时钟仲裁和延展、电气时序和电压规格以及上拉电阻器计算。

## 内容

**1 I²C 概述**....3
**2 I²C 物理层**....4
**3 I²C 协议**....7
**4 I²C 示例**....9
**5 保留地址**....15
**6 高级主题**....20
**7 与 I²C 类似的协议**....31
**8 总结**....31

## 插图清单

图 2-1. 典型的 I²C 实现....4
图 2-2. 开漏连接在 NMOS 开启时将线路拉至低电平....5
图 2-3. 当 NMOS 关闭时，上拉电阻器将线路拉至高电平....5
图 2-4. 开漏和推挽争用之间的比较....6
图 3-1. I²C START 和 STOP....7
图 3-2. I²C 数字一和零表示....7
图 3-3. I²C 地址和数据帧....8
图 4-1. DAC80501 功能框图....9
图 4-2. DAC80501 的 DAC 数据寄存器的示例写入....10
图 4-3. ADS1115 功能框图....11
图 4-4. 配置寄存器....12
图 4-5. I²C 传输，用于从 ADS1115 读取....13
图 4-6. ADC 转换寄存器内容....14
图 5-1. 通用调用地址格式....15
图 5-2. I²C 高速控制器代码....16
图 5-3. 启用 I²C 高速模式....17
图 5-4. I²C 器件 ID 数据位....17
图 5-5. 读取 I²C 器件 ID....18
图 5-6. I²C 十位目标寻址写入....18
图 5-7. I²C 10 位寻址读取....19
图 6-1. 与多个控制器的 I²C 总线争用....20
图 6-2. I²C 时钟同步 SCL 变为低电平....21
图 6-3. I²C 时钟同步 SCL 返回高电平....21
图 6-4. I²C 时钟同步监控 SCL....22
图 6-5. I²C 时钟同步产生有线 And....22



English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

图 6-6. I²C 控制器仲裁.....23
图 6-7. I²C 目标时钟延展.....24
图 6-8. 总线电压不匹配时的 I²C 上拉电阻器驱动下输入.....26
图 6-9. 总线电压不匹配时的 I²C 上拉电阻器过驱动输入.....27
图 6-10. 在耐受更高电压的线路上对 I²C 上拉电阻器进行过驱动.....27
图 6-11. PCA3906 I²C 电压电平转换器.....28
图 6-12. 影响确定上拉电阻器大小的因素.....28
图 6-13. 基于下拉电流的最小上拉电阻.....29
图 6-14. 基于上拉电阻上升和总线电容的最大上拉电阻器.....30

## 表格清单

表 1-1. 不同 I²C 模式的最大传输速率.....3
表 5-1. 保留 I²C 地址列表.....15
表 6-1. 有线 AND 真值表.....20
表 6-2. ADS1119 数字输入和输出的电气特性.....25
表 6-3. I²C 协议的参数特性.....29
表 6-4. SDA 和 SCL 输入和输出电压的特性.....29

## 商标

SMBus® is a registered trademark of System Management Interface Forum, Inc..
PMBus)® is a registered trademark of Intel.
所有商标均为其各自所有者的财产。



English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

# 1 I²C 概述

I²C 是使用串行数据线 (SDA) 和串行时钟线 (SCL) 的双线串行通信协议。该协议支持通信总线上的多个目标器件，也可以支持多个发送和接收命令与数据的控制器。通信以字节数据包形式发送，每个目标器件具有唯一的地址。

## 1.1 发展历程

I²C 通常被称为 I ‘two’ C，代表内部集成电路协议。I²C 由 Philips Semiconductor ( 现为 NXP Semiconductor ) 于 1982 年开发，作为一种低速通信协议，用于将微控制器和处理器等*控制器* 器件与数据转换器和其他外设器件等*目标* 器件连接起来。自 2006 年以来，实现 I²C 协议无需许可证，包括 TI 在内的许多半导体器件公司都推出了与 I²C 兼容的器件。

出于多种原因，I²C 协议被广泛使用。该协议只需要两条通信线路。与其他串行通信协议一样，它也有一条串行数据线和一条串行时钟线。I²C 只能通过两条线连接到总线上的多个器件。控制器器件可以通过串行数据线发送的唯一 I²C 地址与任何目标设备通信。对于器件制造商而言，实现 I²C 既简单又经济。

## 1.2 I²C 速度模式

I²C 有多种速度模式，从*标准模式 (Sm)* 开始，该模式是一种运行速度高达 100Kb/s (kbps) 的串行协议。此模式之后是*快速模式 (Fm)*，其最高速度为 400kb/s。如果总线电容和驱动能力允许更快的速度，则控制器可以使用快速模式。这两种协议都得到广泛支持。

*超快速模式 (Fm+)* 模式可实现高达 1 兆位/秒 (Mbps) 的通信。为了实现这一速度，器件中的驱动器需要额外的强度来满足更短的上升和下降时间要求。

这三种模式相对相似，使用的通信结构是相同的。但是，对于每种模式，所有器件中 I²C 的时序规格都不同，并且硬件实现也不同，以适应不同的速度。

I²C 还具有其他两种用于更高数据速率的模式。*高速模式 ( Hs 模式 )* 的数据速率高达 3.4 兆比特每秒。在此模式下，控制器器件必须首先使用控制器代码来实现高速数据传输。这将在目标器件中启用高速模式。该模式还需要有源上拉电阻以更高的数据速率驱动通信线路。

*超快速模式 (UFm)* 是最快的工作模式，可传输高达 5Mbps 的数据。该模式为只写模式，在通信协议中省略了一些 I²C 功能。

表 1-1 显示了不同的 I²C 模式及其各自的数据速率

**表 1-1. 不同 I²C 模式的最大传输速率**

| I²C 模式 | 最大比特率 |
|---|---|
| 标准模式 | 100kbps |
| 快速模式 | 400kbps |
| 超快速模式 | 1Mbps |
| 高速模式 | 3.4Mbps |
| 超快速模式 | 5Mbps |


English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

# 2 I²C 物理层

## 2.1 两线制通信

I²C 系统为总线上的所有器件提供两条共享通信线路。这两条线路用于双向半双工通信。I²C 支持多个控制器和多个目标器件。这两条线路都需要上拉电阻器。图 2-1 展示了 I²C 物理层的典型实现。

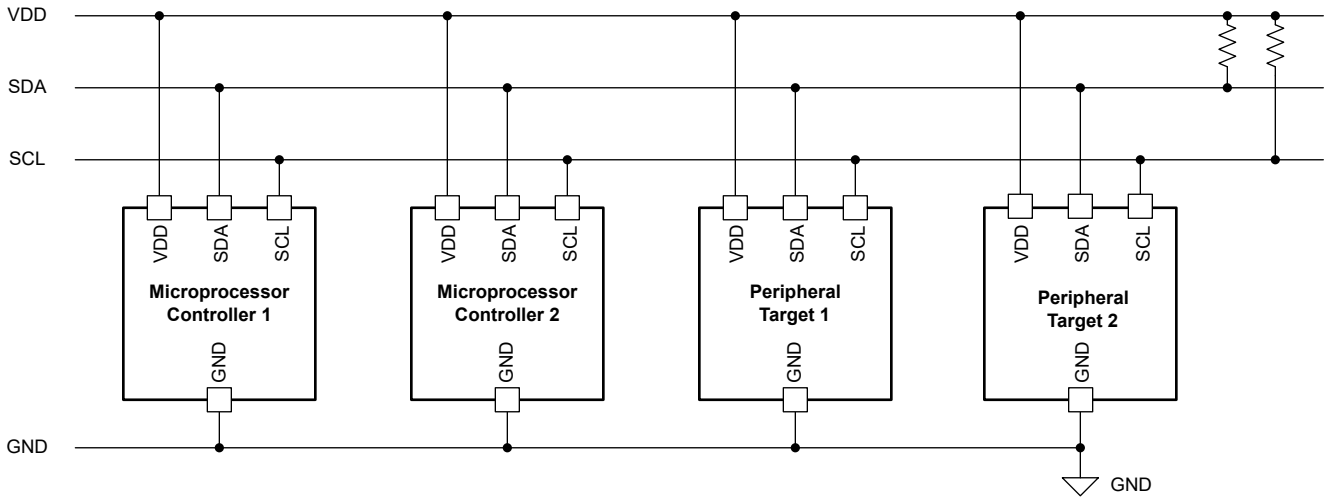



**图 2-1. 典型的 I²C 实现**

I²C 采用通用协议的原因之一是因为它只有两条通信线路。第一条线是 SCL，它是主要由控制器器件控制的串行时钟。SCL 用于同步目标器件的时钟数据输入或输出。第二条线路是 SDA，它是一条串行数据线。SDA 用于将数据传输到目标器件或从目标器件传输数据。例如，控制器器件可以将配置数据和输出代码发送到目标数模转换器 (DAC)，或者目标模数转换器 (ADC) 可以将转换数据发送回控制器器件。

I²C 是半双工通信，其中一次只有单个控制器或目标器件在总线上发送数据。相比之下，串行外设接口 (SPI) 是一种全双工协议，在这种协议中，可以同时将数据发送和接收回来。SPI 需要四条通信线路，使用两条数据线路与目标器件之间发送数据。除了串行时钟之外，独特的 SPI 片选线路选择器件进行通信，有两条数据线路用于目标器件的输入和输出。

I²C 控制器器件启动和停止通信，从而消除潜在的总线争用问题。与目标器件的通信通过总线上的唯一地址发送。这允许 I²C 总线上有多个控制器和多个目标器件。

SDA 和 SCL 线路与总线上的所有器件都有开漏连接。这需要将一个上拉电阻器连接至公共电压电源。



English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

## 2.2 开漏连接

SDA 和 SCL 线路上均使用开漏连接，并连接到 NMOS 晶体管。该开漏连接控制 I²C 通信线路并将线路拉至低电平或者释放线路至高电平。开漏是指当 NMOS 关闭时的 NMOS 总线连接。图 2-2 展示了 NMOS 导通时的开漏连接。

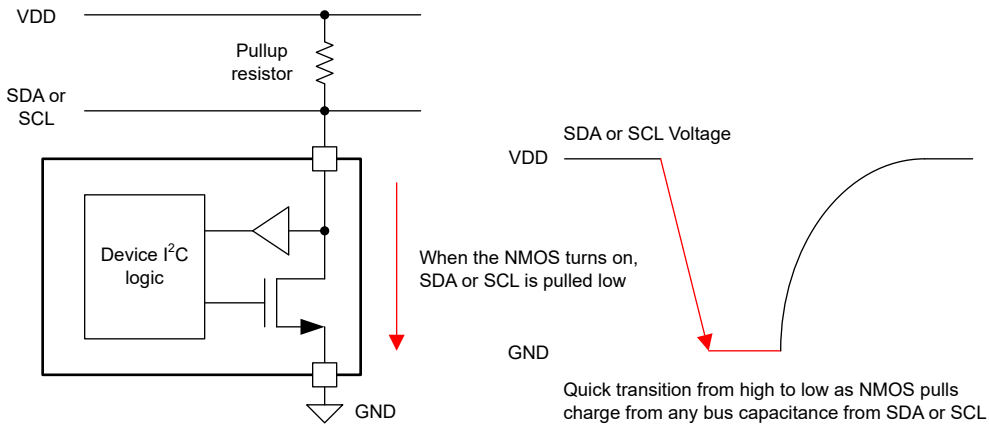



**图 2-2. 开漏连接在 NMOS 开启时将线路拉至低电平**

要设置 SDA 或 SCL 线路的电压电平，需设置 NMOS 开启或关闭。当 NMOS 导通时，器件通过电阻器将电流拉至地。这会将开漏线路拉至低电平。通常，I²C 从高电平到低电平的转换是在 NMOS 在 SDA 或 SCL 上下拉时的快速转换。转换速度由 NMOS 驱动强度以及 SDA 或 SCL 上的任何总线电容决定。

当 NMOS 关断时，器件停止拉电流，上拉电阻器将 SDA 或 SCL 线路拉至 VDD。图 2-3 显示了 NMOS 关闭时的开漏线路。上拉电阻器将线路拉至高电平。开漏线路的转换较慢，因为线路针对总线电容上拉，并且未被主动驱动。

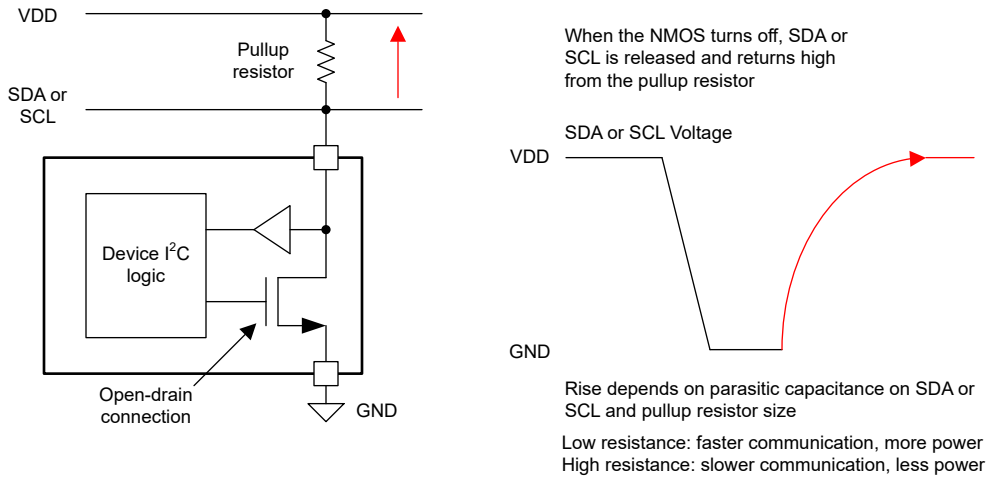



**图 2-3. 当 NMOS 关闭时，上拉电阻器将线路拉至高电平**

通过控制该开漏连接，SDA 和 SCL 均可设置为高电平和低电平，从而启用 I²C 通信。

由于 I²C 通信线上的电容，SDA 和 SCL 线路以指数建立 RC 时间常数放电，具体取决于 I²C 总线上的上拉电阻器和电容的大小。较高的电容会限制 I²C 通信速度，器件数量以及总线上器件之间的物理距离。上拉电阻越小，上升时间越短，但通信需要更大的功率。较大的上拉电阻器具有较慢的上升时间，导致通信速度变慢，但所需的功耗更低。


English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

## 2.3 无损总线争用

使用漏极开路 I²C 的优势之一是总线争用不会使总线进入破坏性状态。借助开漏输出，许多器件可以连接在一起而不会产生破坏性争用。对于该连接上的任何输出，如果任何输出将线路拉至低电平，则线路处于低电平。这种连接称为*有线 AND* 连接。当所有输出连接在一起时，输出是所有输出的逻辑 AND。

如果输出是推挽式，则无法将输出连接在一起，否则可能会出现破坏性状态。推挽输出（通常用于 SPI 通信）具有互补的 NMOS 和 PMOS 晶体管，可将输出驱动为高电平或低电平。图 2-4 显示了开漏输出与推挽输出在争用情况下的比较。

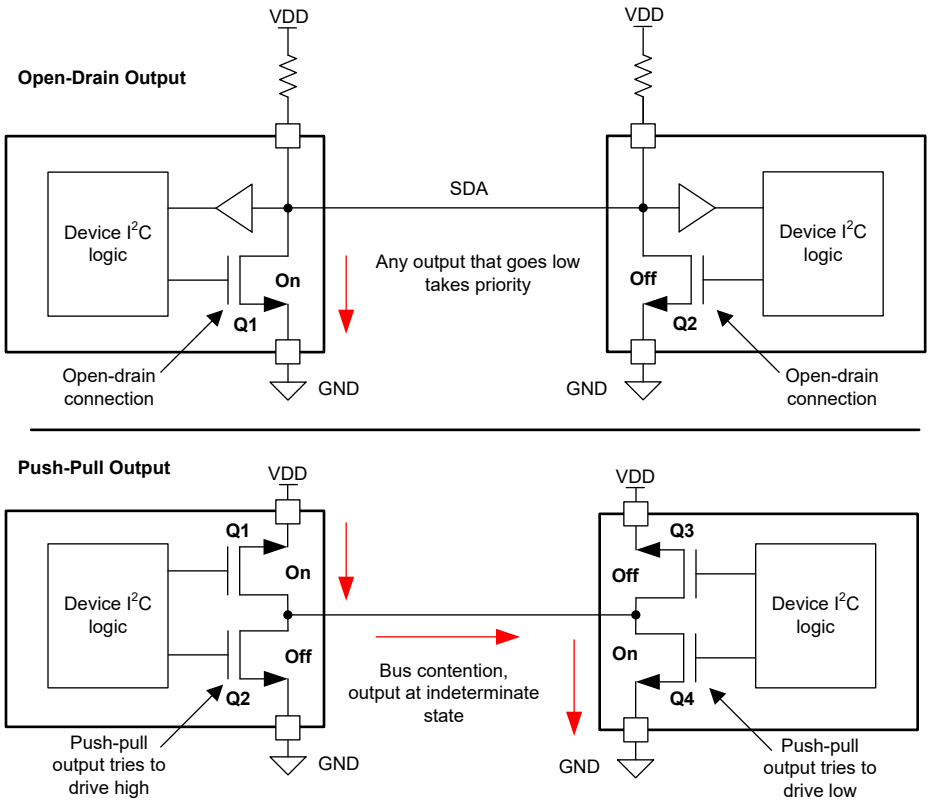



**图 2-4. 开漏和推挽争用之间的比较**

使用开漏连接时，任何器件都可以随时将该连接拉至低电平。只要任何器件将线路拉至低电平，该线路就会显示为低电平，但不会显示为破坏性争用。

在推挽输出中，输出也连接在一起。如果总线上有两个器件处于激活状态，而一个输出为高电平，而另一个输出为低电平，则该总线争用的状态为不确定，可能在中点电压趋稳。此外，一个器件具有 NMOS 传导电流，另一个器件具有 PMOS 传导电流。这些器件通过一条阻抗极低的路径从 $V_{DD}$ 到 GND 拉取电流，从而在晶体管允许的范围内传导尽可能多的电流。该争用会产生大量电流，可能会损坏器件。



English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

# 3 I²C 协议

## 3.1 I²C START 和 STOP

通过 I²C START 条件启动 I²C 通信。如果总线开路，I²C 控制器会通过发送 I²C START 来请求总线进行通信。为此，控制器器件首先将 SDA 拉至低电平，然后将 SCL 拉至低电平。此序列表明控制器器件正在申请 I²C 总线进行通信，从而强制总线上的其他控制器器件保持通信。

当控制器器件完成通信时，SCL 释放为高电平，然后 SDA 释放为高电平。这表明存在 I²C STOP 条件。这样可以释放总线以允许其他控制器进行通信或允许同一控制器与另一器件通信。图 3-1 展示了 I²C START 和 STOP 的协议。

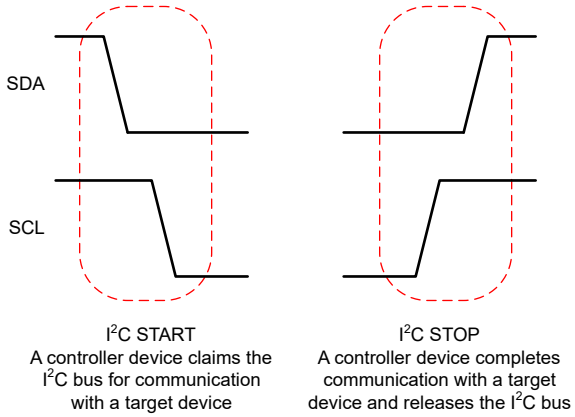



**图 3-1. I²C START 和 STOP**

## 3.2 逻辑 1 和 0

I²C 使用 1 和 0 的序列进行串行通信。SDA 用于数据位，而 SCL 是对位序列进行计时的串行时钟。当 SDA 释放线路时发送逻辑 1，从而允许上拉电阻将线路拉至高电平。当 SDA 下拉线路时发送逻辑零，将低电平设置为接近地电平。图 3-2 展示了用于 I²C 通信的数字 1 和 0 的表示。

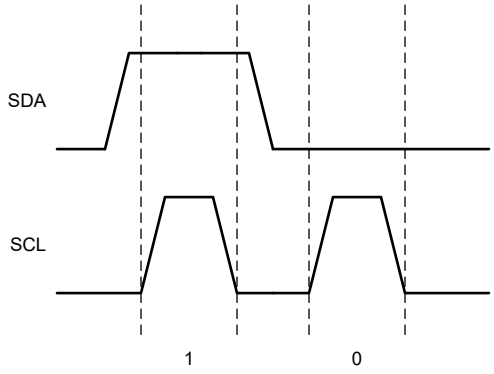



**图 3-2. I²C 数字一和零表示**

当 SCL 出现脉冲时，接收到 1 和 0。对于有效位，SDA 在该位的 SCK 上升沿和下降沿之间不会变化。SCL 上升沿和下降沿之间的 SDA 变化可以解释为 I²C 总线上的 START 或 STOP 状态。


English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

## 3.3 I²C 通信帧

I²C 协议被划分为多个帧。通信从控制器器件在 START 后发送地址帧开始。地址帧后跟一个或多个数据帧，每个数据帧由一个字节组成。每个帧还有一个确认位，用于提醒控制器目标器件或控制器器件已接收通信。图 3-3 显示了两个 I²C 通信帧的图。

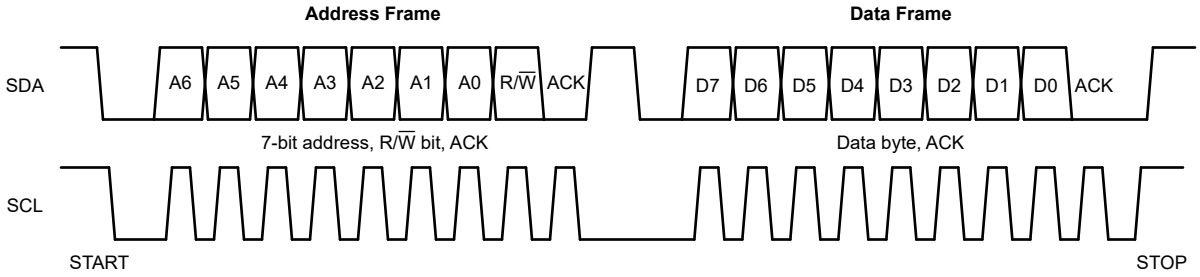



**图 3-3. I²C 地址和数据帧**

在地址帧的开头，控制器器件启动一个 START 条件。控制器器件首先将 SDA 拉至低电平，然后再将 SCL 拉至低电平以用于 START。这使得控制器器件可以在不与总线上其他控制器器件争用的情况下声明总线。每个 I²C 目标器件都有一个关联的 I²C 地址。当开始与特定目标器件通信时，控制器使用目标器件地址在以下 I²C 帧中发送或接收数据。I²C 地址由 I²C 总线上的 7 位器件和器件组成，每个器件在总线上具有唯一的地址。

7 位地址表示 $2^7$（或 128）个唯一地址。但是，有几个保留的 I²C 地址限制了可能器件的数量。节 5 中讨论了保留地址。地址以 SDA 作为数据，SCL 作为串行时钟发送。利用这些信息，您可以读取器件的 I²C 通信，并了解控制器器件和目标器件之间来回发送的内容。

该帧在地址之后的第 8 位是读写 (R/$\overline{W}$) 位。如果该位为 1，则控制器请求从目标器件读取数据。如果该位为 0，则控制器要求将数据写入目标器件。

在任何通信字节之后，会使用额外的第 9 位来验证通信是否成功。在地址字节通信结束时，目标器件在 SCL 脉冲期间下拉 SDA，以向控制器指示已接收到地址。这称为确认 (ACK) 位。如果该位为高电平，则没有目标器件接收到地址并且通信失败。如果该位为高电平，则这称为 NACK，没有 ACK。

地址帧后跟一个或多个数据帧。这些帧每次发送一个字节。在传输每个数据字节后，会有另一个 ACK。如果数据字节是对器件的写入，则目标器件会将 SDA 拉至低电平以对传输进行 ACK。如果数据字节从器件读取，控制器会将 SDA 拉至低电平以确认已接收到数据。ACK 是一个有用的调试工具。该位的缺失可能表示目标外设没有接收到正确的 I²C 通信地址，或者控制器外设没有接收到预期数据。

通信完成后、控制器发出 I²C STOP 条件。SCL 和 SDA 依次被释放。控制器使用 STOP 指示通信已完成且 I²C 总线已释放。

这是控制器器件与目标器件之间任何 I²C 通信的基本协议。通信可以包含多个字节的数据。在某些情况下，目标器件具有多个数据和配置寄存器，从器件读取可以先写入器件，以指示要读取哪个寄存器。以下各部分举例说明了如何对不同的数据转换器器件进行读取和写入。



English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

# 4 I²C 示例

本节使用两个示例来展示 I2C 如何与不同数据转换器进行通信。首先，I²C 协议用于写入 DAC80501 的 DAC 数据寄存器，以设置输出电压。其次，I²C 用于从 ADS1115 ADC 读取转换寄存器。

## 4.1 DAC80501 示例

DAC80501 是一个具有内部基准的 16 位精度电压输出 DAC。如果 SPI2C 引脚在上电时设置为高电平，则该器件使用 I²C 接口并支持 I²C 模式，例如标准模式、快速模式和超快速模式。图 4-1 展示了器件的功能方框图。

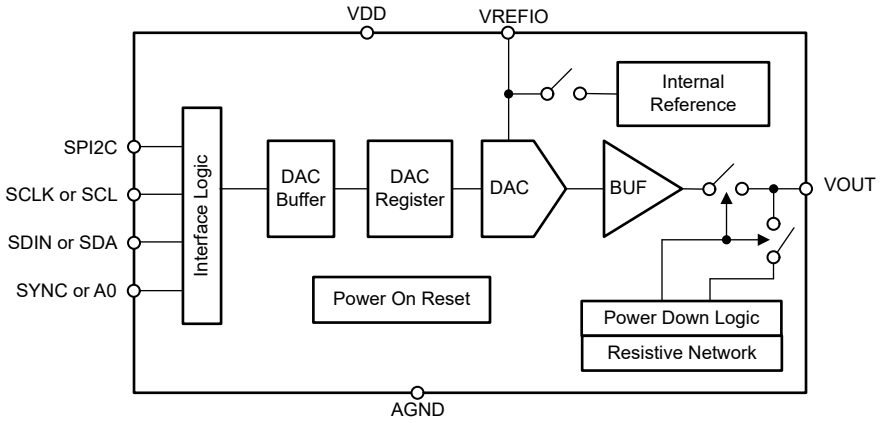



**图 4-1. DAC80501 功能框图**

DAC80501 有一个标记为 A0 的地址引脚。此引脚用于选择四个 I²C 地址之一，这意味着只要器件编程为不同的地址，这些器件中的四个就可以在同一总线上使用。当 A0 引脚连接到 AGND、$V_{DD}$、SDA 或 SCL 时，可以将器件设置为所示的四个唯一地址。在本示例中，A0 设置为 $V_{DD}$，因此地址为 49h。

该器件具有一组寄存器，可用于启用 DAC 基准和输出；通过基准分频器或缓冲器增益设置输出范围；设置复位或启用 LDAC 触发器；以及 DAC 输出代码。还可以读取这些寄存器以验证设置，从不同版本的 DAC 分辨率和上电复位值中识别器件，以及根据基准检查低电源警报。

DAC80501 具有七个内部寄存器，每个寄存器通过命令字节寻址（附加寄存器是一个 NOOP 寄存器（无操作），它是只写寄存器，不发送命令或设置寄存器值）。每个寄存器都有 16 个数据位，其中包含两个用于访问的数据字节。在本示例中，与命令或寄存器字节通信是具有多个寄存器的 I²C 器件中使用的一种常用方法。

DAC 数据寄存器 (08h) 设置 DAC 输出电压。在此示例中，使用 I²C 协议写入 DAC 数据寄存器，以设置器件的输出电压。数据写入由地址字节、寄存器字节和两个数据字节组成，总共四个通信帧。

### *4.1.1 DAC80501 DAC 数据寄存器*

根据方程式 1 中的传递函数通过 DAC 数据寄存器设置 DAC80501 的输出。

$$V_{OUT} = \frac{DAC_DATA}{2^N} \times \frac{VREFIO}{DIV} \times GAIN \quad (1)$$

其中

- DAC_DATA 是 DAC 数据寄存器中的 16 位值
- N 是 DAC 的位数（16 位）
- VREFIO 是基准电压（2.5V 内部基准）
- DIV = 1（默认设置）
- GAIN = 2（默认设置）

例如，DAC 输出可设为 1.5V 的输出电压。为此，请根据传递函数公式计算电压。将 VOUT 设置为 1V 并计算 DAC_DATA 的值。


English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

$$\text{DAC_DATA} = \frac{V_{OUT}}{GAIN} \times \frac{DIV}{VREFIO} \times 2^{N} = \frac{1.5\ V}{2} \times \frac{1}{2.5\ V} \times 2^{16} = 19661 = 4CCDh \quad (2)$$

要将 4CCDh 写入 DAC80501，需要写入四个字节。I²C 通信从地址字节开始，然后是命令字节，以指示要写入哪个寄存器。通信帧以 DAC_DATA 寄存器的两个字节结束，用于设置输出电压。

### 4.1.2 DAC80501 I²C 示例写入

I²C 写入从 START 条件开始。SDA 被设置为低电平，然后 SCL 被设置为低电平。然后写入 I²C 地址。在 A0 引脚连接到 VDD 的情况下，DAC80501 会响应地址 100 1001（或 49h）。读/写 (R/$\overline{W}$) 位设置为低电平，表示控制器正在写入设备。

地址字节完成后，DAC80501 通过下拉地址帧最后一位的 SDA 来对地址进行 ACK。控制器向总线上的所有目标器件发送地址和读取或写入信息。如果目标 DAC80501 具有匹配的地址，器件会发送一个 ACK 以向控制器指示接收到有效地址，从而器件已准备好接收信息。

控制器发送写入 DAC80501 的地址后，控制器会告知器件正在写入哪个寄存器。发送至目标器件的第二个字节是 DAC 数据寄存器的寄存器指针。这里，将 0000 1000 发送到 DAC80501。作为响应，DAC80501 将 SDA 设置为低电平以进行 ACK。同样，目标器件向控制器指示器件已接收到地址指针数据以及正在写入哪个寄存器。

现在，DAC 数据寄存器值每次发送到目标器件一个字节。对于此字节，发送 DAC 数据的第一字节。位 0100 1100 发送到 DAC80501。DAC80501 会对第一个字节进行 ACK。

最后，配置寄存器的最后一个字节发送到目标器件。这里，将 1100 1101 发送到 DAC80501。DAC80501 会对第二个数据字节进行 ACK。最后，SCL 被释放为高电平，然后 SDA 被释放。在此操作中，控制器通过发出 STOP 条件来释放总线。

将这些帧放在一起后，I²C 写入显示为图 4-2。在这里，该图显示了针对所有帧采用正确位设置的整个通信。如果示波器绘制了 SDA 和 SCL 的 I²C 通信图，则可以直接将此图与该图进行比较以进行调试。

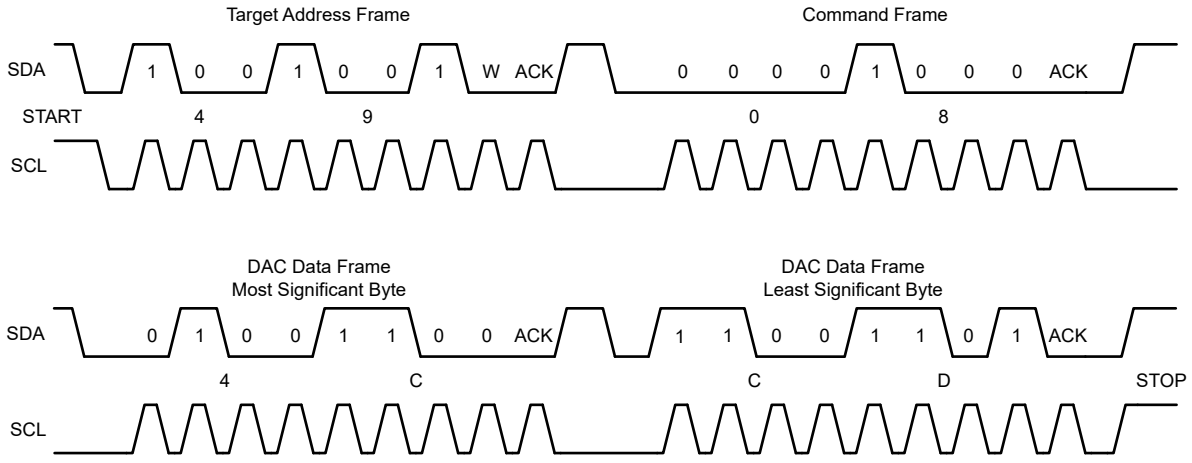



**图 4-2. DAC80501 的 DAC 数据寄存器的示例写入**



English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

## 4.2 ADS1115 示例

在第二个示例中，使用 I²C 从精密 ADC 读取数据。ADS1115 是一款使用 I²C 接口的 16 位精密 ADC，可支持标准、快速和高速模式。该器件具有可通过配置寄存器进行设置的多种设置，包括可编程增益放大器（或 PGA）设置的输入范围、各种数据速率，以及可设置为进行差分测量或相对于接地的单端测量的输入多路复用器。图 4-3 展示了 ADS1115 的功能方框图。

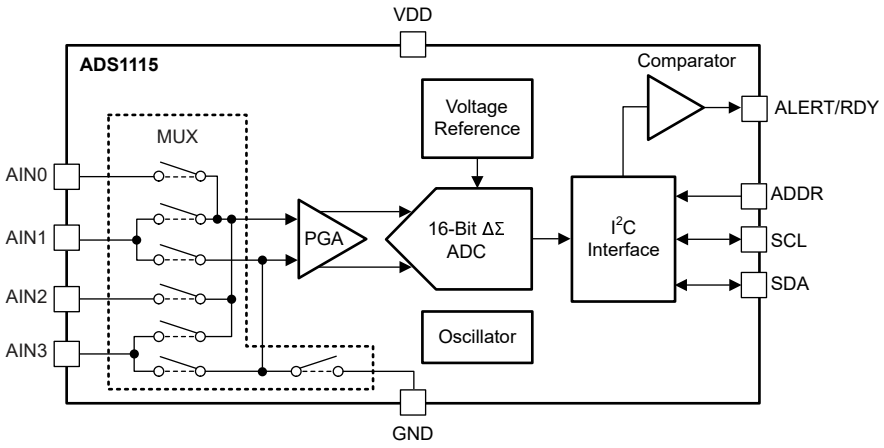



**图 4-3. ADS1115 功能框图**

与前一个示例类似，ADS1115 有一个标记为 ADDR 的地址引脚。此引脚用于选择四个 I²C 地址之一，这意味着只要器件编程为不同的地址，这些器件中的四个就可以在同一总线上使用。用于该器件的 I²C 地址取决于 ADDR 引脚连接。当 ADDR 引脚连接到 VDD、SDA 或 SCL 时，该器件可以设置为表中所示的其他地址。在本示例中，ADDR 设置为接地，因此地址设置为 48h。

ADS1115 具有四个内部寄存器，每个寄存器由一个内部指针寄存器进行寻址。第一个寄存器是转换数据寄存器。当 ADC 完成转换时，将 ADC 数据放入该寄存器中，控制器器件会读取该寄存器。

第二个寄存器是配置寄存器。控制器器件向该寄存器写入值以对器件进行编程并开始转换。配置寄存器设置器件的可编程增益放大器、输入通道、数据速率和其他工作模式。

最后两个寄存器是 Lo_threshold 寄存器和 Hi_threshold 寄存器。这两个寄存器用于设置器件中数字比较器的阈值。一旦转换数据超过这些阈值，器件就可以为 ALERT/RDY 引脚设置警报。在本示例中，未配置比较器和阈值。

在此示例中，首先显示配置寄存器的设置来说明 ADC 的设置和范围。之后，显示了 ADC 从转换寄存器读取的示例 I²C。


English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

### 4.2.1 ADS1115 配置寄存器

ADS1115 有一个 16 位配置寄存器。写入该寄存器对器件的配置进行编程并开始转换。本部分详细介绍了这些设置以及器件的运行模式。图 4-4 显示了配置寄存器数据字段。这显示了配置寄存器中的功能及其位的位置。确定配置寄存器的所有设置后，使用 I²C 写入来设置寄存器。

**图 4-4. 配置寄存器**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| OS | MUX[2:0] | | | PGA[2:0] | | | 模式 |
| R/W-1h | R/W-0h | | | R/W-2h | | | R/W-1h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DR[2:0] | | | COMP_MODE | COMP_POL | COMP_LAT | COMP_QUE[1:0] | |
| R/W-4h | | | R/W-0h | R/W-0h | R/W-0h | R/W-3h | |

图 4-4 示出了配置寄存器字段说明，对位设置进行了详细说明。从最高有效位开始，第 15 位是单次转换开始位。

位 14 至 12 设置器件的多路复用器设置。在该示例中，器件设置为从 AIN0 对地进行单端测量。在配置寄存器中，将 AINP 设置为 AIN0，AINN 设置为 GND。为此，请将 14 至 12 位设置为二进制的 100。

第 11 到 9 位用于配置器件的 PGA 设置。这是可编程增益放大器的设置。这将设置输入测量的满量程，设置 ADC 可以测量的输入信号大小。设置 ADC 以测量正负 4.096V 的大信号。将第 11 至 9 位设置成二进制的 001。

第 8 位用于设置器件的工作模式。对于此操作，将器件设置为单次转换模式，将第 8 位设置为 1。

第 7 至 5 位用于设置器件 ADC 的数据速率。将此位设置为最高数据速率，即每秒 860 个样本。将第 7 到 5 位设置为 111。

最后五位（从 4 到 0）控制该器件的数字比较器。不使用数字比较器，并通过寄存器的最后两位禁用该比较器。其余位处于其默认设置。将第 4 到 0 位设置为 00011。

ADS1115 的配置寄存器设置到此结束。这些位用于对寄存器进行写入。该寄存器也可以用十六进制表示为 C3E3h。要将其写入器件，请使用写入目标器件时所用的相同格式，对于 DAC80501，请参阅图 4-2。



English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

### 4.2.2 ADS1115 I²C 示例读取

ADS1115 具有 16 位 ADC，因此可以进行 16 位数据转换。控制器器件从转换寄存器读取以获取 ADC 转换数据。转换寄存器地址指针为 00h。转换数据以 16 位结果形式显示为二进制补码。正满量程输入可生成输出代码 7FFFh，负满量程输入可生成输出代码 8000h。

当 ADDR 引脚连接到 GND 时，器件响应地址 48h。图 4-5 显示了从 00h 地址指针处的转换数据寄存器读取的示例。

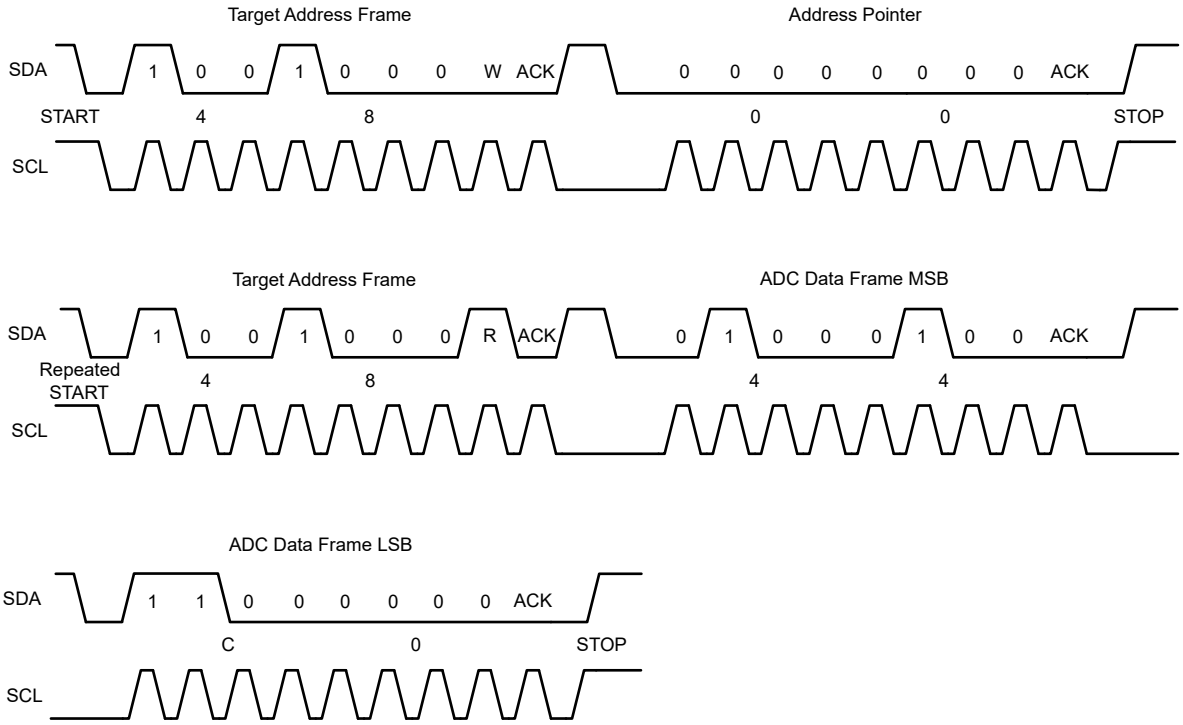



**图 4-5. I²C 传输，用于从 ADS1115 读取**

I²C 写入从 START 条件开始。SDA 被设置为低电平，然后 SCL 被设置为低电平。然后，控制器写入 I²C 地址。同样，器件响应地址 100 1000（或 48h）。

控制器首先需要告诉器件要读取哪个寄存器。在此情况下，通信首先向器件发送 I²C 写入，以便设置从 ADS1115 的数据转换寄存器进行读取。此时，R/$\overline{W}$ 位被设置为低电平，表示通信从写入器件开始。

地址帧完成后，ADS1115 通过拉低地址帧最后一位的 SDA 来对地址进行 ACK。

指示控制器正在从 ADS1115 读取数据后，控制器会告知器件从哪个寄存器读取。第二个字节是数据转换寄存器的寄存器指针。在这里，将 00h 发送到 ADS1115。作为响应，ADS1115 将 SDA 设置为低电平以进行 ACK。最后，控制器发出 STOP 命令以释放总线。

现在控制器已通知器件访问数据转换寄存器，控制器随后发出从寄存器中读取的指令。控制器再次写入 I²C 地址。控制器已经指示要读取哪个寄存器，现在控制器向器件发送读取信息，以便可以读取 ADS1115 的数据转换寄存器。此时，R/$\overline{W}$ 位设置为高电平，表示读取。同样，地址帧完成后，ADS1115 会对地址进行 ACK。

接下来的两个数据字节用于读取寄存器。第一个字节是转换数据的最高有效字节，然后第二个字节作为对最低有效字节转换数据的读取。每个字节的控制器随后都有一个来自控制器的 ACK。最后，控制器发送 STOP 以结束 I²C 通信。

与上一个示例一样，图 4-5 是一个方便的图，显示了与器件的 I²C 通信。如果通信存在问题，可以使用 SDA 和 SCL 的示波器图与该示例写入进行比较。


English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

### 4.2.3 ADS1115 转换结果

使用转换寄存器结果，该示例继续计算转换结果。转换寄存器读数为 44C0h，即十进制的 17600。这是基于测量输入电压的 ADC 输出。图 4-6 展示了转换寄存器的数据内容。

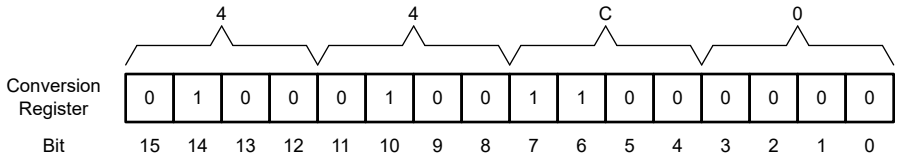



**图 4-6. ADC 转换寄存器内容**

使用转换寄存器中的值，将转换寄存器转换为用于 ADC 测量的电压。正满标量程为 4.096V 时，将此数据转换为测量的电压。

$$\text{Conversion Voltage} = 4.096\ \text{V} \times \text{Data Code} / 2^{15} = 4.096\ \text{V} \times 17600 / 32768 = 2.2\ \text{V} \quad (3)$$

ADC 数据代码为 17600d 时，ADC 会报告 2.2V 的测量值。



English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

# 5 保留地址

在 I²C 协议中，对目标器件进行写入和读取需要使用 I²C 地址。I²C 地址标识控制器要与哪个器件通信。通常，该地址是通过单个字节写入的，其中地址本身为 7 位，而第八个额外的位用于指示对器件的读取或写入

不过，并非 7 位地址的所有地址都可用于目标器件。某些地址被保留用于其他目的。本节介绍这些保留地址用于哪些功能。

I²C 有几组保留地址，这些地址仅限于根据特定应用使用。这些保留地址调用的函数是器件的选项，并非所有 I²C 器件都可用。表 5-1 列出了这些保留地址的一组及其功能。

**表 5-1. 保留 I²C 地址列表**

| 目标地址 | R/$\overline{W}$ 位 | 说明 |
|---|---|---|
| 000 0000 | 0 | 常规调用地址 |
| 000 0000 | 1 | START 字节 |
| 000 0001 | X | C-Bus 地址 |
| 000 0010 | X | 为不同的总线格式保留 |
| 000 0011 | X | 保留以供将来使用 |
| 000 01XX | X | Hs 模式控制器代码 |
| 111 11XX | 1 | 器件 ID |
| 111 10XX | X | 10 位目标地址 |

## 5.1 常规调用

第一个保留地址是 I²C 地址 0，是通用广播地址。对通用广播地址的写入操作用于同时对连接到 I²C 总线的所有器件进行寻址。并非所有器件都设计为对应通用呼叫地址。但是，如果有响应，则目标器件可以在通用调用之后处理第二个字节和后续字节。图 5-1 显示了通用调用地址的两字节格式。

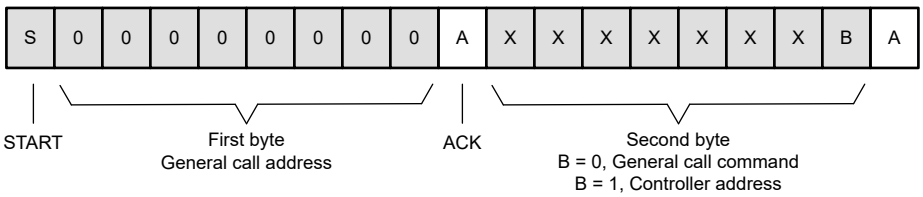



**图 5-1. 通用调用地址格式**

器件的响应有两种不同的情况，具体取决于通用广播之后字节的最低有效位。如果第二个字节 (B) 的最低有效位为零，则第二个字节可用于向接收通用广播的器件发送命令。例如，通过 I²C 发送的一个常用命令是通用广播复位。如果第二个字节为 06h，则控制器器件向 I²C 总线上的所有器件发送通用广播复位。

如果 B 是 1，则控制器作为硬件通用广播发送两字节序列。在这种情况下，控制器器件可以是硬件控制器，它无法编程为传输到特定目标地址，或者不知道将命令发送到哪个器件。在这种情况下，第二个字节用于发送控制器地址，以便向系统上的所有器件标识其自身。然后，系统中的另一个控制器器件可识别此地址，该控制器器件可用于将信息定向到用作目标器件的硬件控制器。


English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

## 5.2 START 字节

从 I²C 地址中读取出的 00 为 START 字节。并非所有微处理器都具有内置的板载 I²C 控制器，不一定具有中断功能以检测总线上的通信。在这种情况下，负责监控 I²C 总线的微处理器必须反复轮询 SDA 和 SCL 线，以确保通信不会丢失。这要求器件以快速轮询速率检测 I²C 地址。

一种选项是在通信之前使用 START 字节。当控制器使用地址中包含 7 个零的 START 字节时，目标器件能够以慢得多的速率轮询，从而节省处理能力。

目标器件检测到 SDA 在 START 字节期间发送零后，该器件可以切换到更快的轮询模式，以检测正在发送地址的下一次 I²C 传输。I²C 传输以 STOP 条件结束后，目标器件随后可以较慢的速率恢复轮询，从而再次节省处理能力。

## 5.3 C-Bus 地址、不同的总线格式、未来用途

表中列出的接下来三个 I²C 地址因不同原因全部得到保留。地址 01 为 C-Bus 协议而保留，以便 C-Bus 器件可以放置在 I²C 总线上。然而，在世界某些地区，C-Bus 协议用于家庭和楼宇自动化，但在美国通常不使用。该 I²C 地址被大多数器件忽略。

地址 02 为不同的总线格式而保留。旨在实现不同协议之间的通信。只有在不同协议之间工作的 I²C 器件才能响应此地址。

保留地址 03 以供未来使用（尚未定义）。

## 5.4 HS 模式控制器代码

接下来的保留地址用于高速控制器代码。这些代码为 04 至 07。通常用于读取或写入指示的第八位用作高速控制器代码的一部分。这些高速控制器代码是保留的 8 位代码，不用于目标寻址或其他目的。每个高速控制器都具有一个唯一的控制器代码，因此在 I²C 总线上最多可支持八个高速控制器。高速模式控制器器件的控制器代码可通过软件编程，并由系统设计人员选择。

支持高速模式的器件开始在标准或快速模式下运行。控制器代码启用高速模式。高速控制器代码允许在高速控制器之间进行仲裁并指示高速模式传输开始。代码启用内部电流源，从而使 I²C 通信总线比仅使用上拉电阻器更快。

启用后，高速数据传输会在数据传输过程中继续进行。重复 START 将继续进行高速模式数据传输，而 STOP 条件将 I²C 总线恢复为快速或标准模式。图 5-2 显示了开始传输到高速控制器代码的情况。该图展示了高速模式传输的开始。

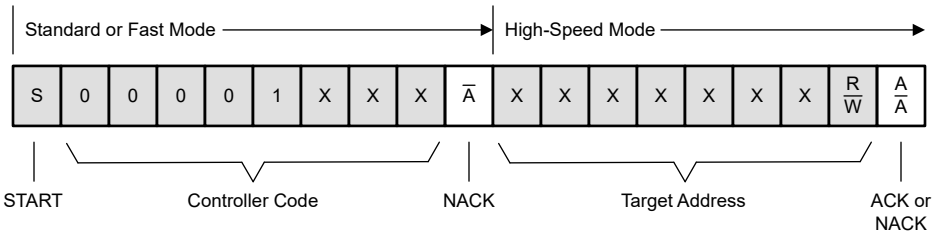



**图 5-2. I²C 高速控制器代码**

在这个及后续的字节和位帧图中，阴影代码由控制器器件设置，而非阴影代码由目标器件设置。



English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

通信开始时，器件以标准或快速模式启动。控制器通过将 SDA 下拉为低电平然后将 SCL 下拉为低电平来发送 START 条件。然后，发送第一个字节以及用于高速控制器代码的保留地址。控制器代码为所有支持高速模式的器件启用高速模式并启用高速模式的控制器的内部电路。图 5-3 展示了一个详细图，说明了如何启用 I²C 高速模式然后用数据进行通信。

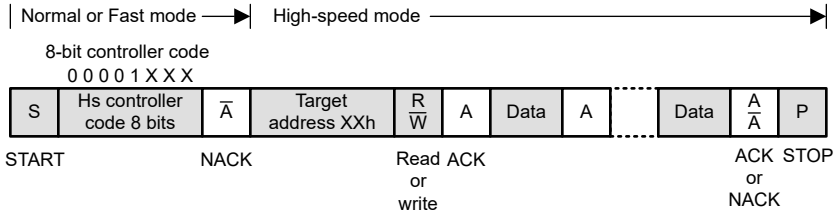



**图 5-3. 启用 I²C 高速模式**

然后，控制器发送高速模式器件的目标地址，并随后发送用于通信的读取或写入位。数据由控制器或目标器件传输，每个数据字节都具有 ACK，类似于标准 I²C 通信。目标器件会继续通信，直到接收到 STOP 条件或接收到新目标地址的重复 START。

## 5.5 器件 ID

地址 7C 至 7F 均为器件 ID 保留。控制器首先发送保留的器件 ID 地址，后跟写入位。然后，控制器发送它想要识别的器件地址。然后，控制器发送一个重复的 START 条件，后跟保留的器件 ID 地址，后跟一个读取位。

器件 ID 由目标器件通过三个 I²C 数据帧中的字节发送。此数据从 12 位制造商 ID 开始，接着是 9 位器件标识，最后是 3 位芯片版本。图 5-4 展示了到器件 ID 的数据传输。

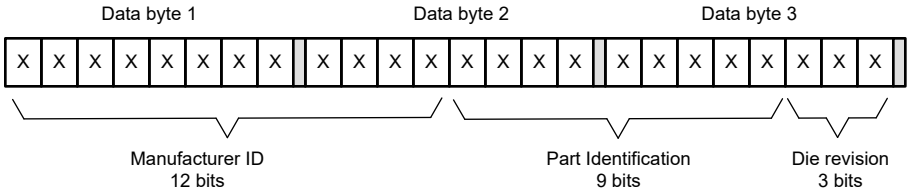



**图 5-4. I²C 器件 ID 数据位**

控制器先发送一个 START 条件。第一个字节与器件 ID 的保留地址一起发送，后跟一个 0 用于写入。此时，可以有多个目标对器件 ID 做出响应，因此多个器件可以对该地址进行 ACK。

然后，控制器发送目标器件的地址。该目标地址字节的最后一位是“无关”，后跟一个 ACK。此时，只有一个器件对该地址进行 ACK。

然后，控制器发送一个重复 START 条件。之后，控制器发送保留的器件 ID I²C 总线地址，后跟读取位。目标器件对该保留地址进行 ACK。请注意，第三个字节的开头必须是重复的 START。STOP 后跟 START 条件，或 STOP 后跟重复 START 条件，然后访问不同的目标器件，都会使目标器件状态机复位，因此无法执行器件 ID 读取。


English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

然后，目标器件发送三个字节作为器件 ID。图 5-5 展示了有关如何从支持 I²C 器件 ID 的器件读取器件 ID 的详细图。

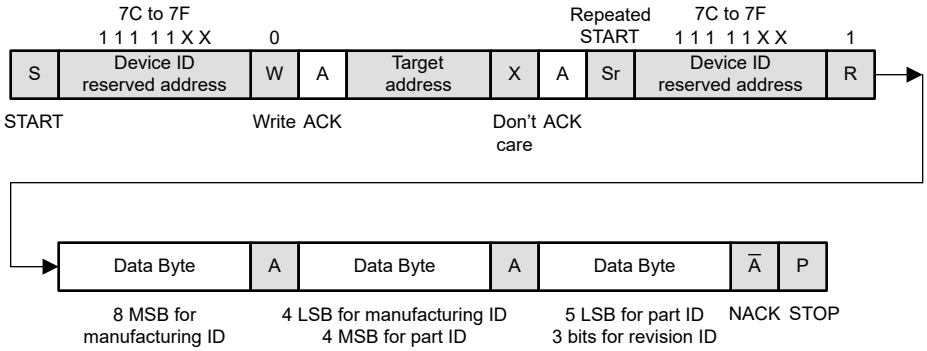



**图 5-5. 读取 I²C 器件 ID**

控制器会对最后一个字节进行 NACK，并以 STOP 结束器件 ID 读取。通过发送 NACK，可以随时停止读取器件 ID。如果控制器在第三个字节之后继续对这些字节进行 ACK，则目标会回滚到第一个字节并持续发送器件 ID 序列，直到检测到 NACK。

## 5.6 10 位目标寻址

使用正常的 7 位 I²C 地址和所有保留地址，总线上可能的 I²C 器件的数量将受到限制。若要增加器件数量，可以使用几个保留地址将地址扩展到 10 位。在保留地址中，78h 至 7Bh 的最后两位表示用于扩展地址空间的前两位。第二个八位的完整字节用于完成 10 位地址。与 10 位地址的通信与 7 位地址的通信非常相似。

### *5.6.1 10 位目标寻址写入*

图 5-6 显示了写入支持 10 位寻址的器件的协议。在 10 位地址写入开始时，控制器发送一个 START 条件。然后，发送第一个字节，其中包含用于 10 位 I²C 寻址的保留地址，后跟一个用于写入的 0。保留地址的最后两位包括 10 位地址的前两位。此时可以有多个目标具有相同的第一个地址字节用于 10 位寻址，因此多个器件可以对该地址进行 ACK。第二个字节包含目标地址。该字节是 10 位目标地址的 8 个最低有效位。

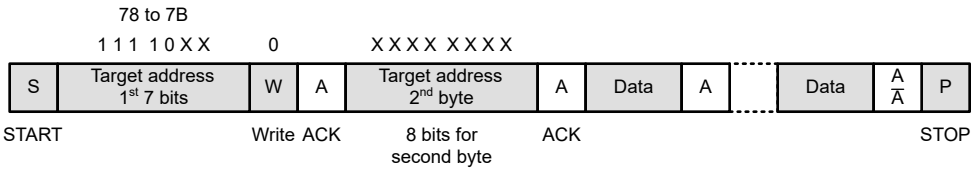



**图 5-6. I²C 十位目标寻址写入**

使用第二个字节时，大概只有一个器件具有唯一的 10 位地址。这是唯一的一个对通信进行 ACK 的器件。使用保留地址的两位和第二个字节目标地址的八位，总共产生 10 位寻址。

此目标器件会保持通信，直到控制器发送 STOP 条件或控制器发送重复的 START 条件来与不同的目标地址通信。



English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

### 5.6.2 10 位目标寻址读取

从 10 位寻址器件读取与写入类似，但增加了步骤。在通信开始时，会出现一个 START，后跟保留地址。然后将写位的 0 写入。随后是所有使用保留地址的器件发出的 ACK。然后发送目标器件地址第二个字节。然后接收来自被寻址器件的 ACK。在这之前，通信与 10 位地址写入完全相同。

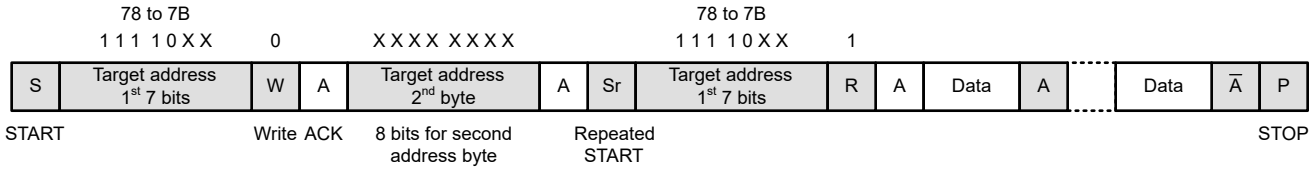



**图 5-7. I²C 10 位寻址读取**

要从该器件读取数据，控制器随后发送一个重复的 START 条件。接下来是刚刚使用的保留地址。然后发送读取位，后跟一个 ACK。由于发送的是读取位（而不是写入位），因此之前对该通信进行 ACK 的器件会解释这是读取。具有相同保留地址的其他设备没有响应。然后，被寻址的器件对该保留目标地址的重复 进行 ACK。

在被寻址的器件发送 ACK 后，器件发送数据，在每个字节之后，控制器都会对数据进行 ACK。此数据传输将继续，直到控制器发送 STOP 条件或重复的 START 条件，后跟不同的目标地址。


English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

# 6 高级主题

在前面几节中，我们通过一些示例来讨论了 I²C 的协议基础知识，以展示与精密数据转换器的通信。这些章节介绍了 I²C 的工作原理以及如何读取、写入和帮助调试基本系统通信。

但是，这些描述只是涉及 I²C 协议的基本内容。本节介绍了 I²C 的一些高级主题。不详细介绍以下内容。但是，本节介绍了一些主题，可让您了解这些主题是什么。更多详细信息，请参阅 I²C 总线规格。

## 6.1 时钟同步和仲裁

此部分的第一个 I²C 主题是总线上控制器器件之间的时钟同步和仲裁。在 I²C 中，同一总线上可以有多个控制器。因此，可能有两个或更多器件同时尝试请求总线进行通信。这需要多个有效控制器来解析由哪个器件控制总线。

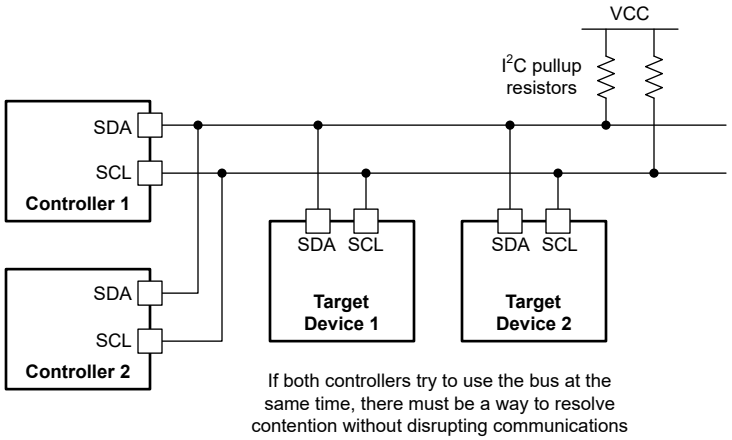



**图 6-1. 与多个控制器的 I²C 总线争用**

I²C 使用一种时钟同步和仲裁方法来确保一个控制器获得控制权，而不会影响通信。由于 I²C 使用与 SDA 和 SCL 的开漏连接，因此该连接会产生线与连接，其中线路提供器件输出的逻辑 AND。这有助于在不中断通信的情况下进行仲裁。在只有一个控制器的系统中，不需要进行该仲裁。

本部分详细介绍了时钟同步以及多个控制器如何为 I²C 同步时钟以防止争用。还介绍了控制器如何利用仲裁来确定哪个控制器在没有中断争用的情况下占用总线。

为防止总线争用，首先使用 SCL 线路和总线上控制器的开漏连接来执行时钟同步。如果任何控制器将 SCL 拉至低电平，则该有线 AND 连接为低电平。该连接是两个控制器器件 SCL 连接的逻辑 AND。仅当两个控制器器件都释放开漏连接为高电平时，SCL 的输出才为高电平。表 6-1 详细说明了这个逻辑有线 AND 的真值表。

**表 6-1. 有线 AND 真值表**

| 控制器 1 SCL | 控制器 2 SCL | 产生的 SCL |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

在两个控制器试图声明总线的 START 条件下，SCL 上存在从高电平到低电平的转换。在以下示例中，两个控制器器件尝试同时或接近同时声明总线。

在图 6-2 中，控制器 1 器件在控制器器件 2 执行相同操作之前不久启动 START 条件。控制器 1 在控制器 2 之前下拉 SCL。在使用有线 AND 连接的情况下，一旦控制器 1 下拉 SCL，SCL 就会拉低。



English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

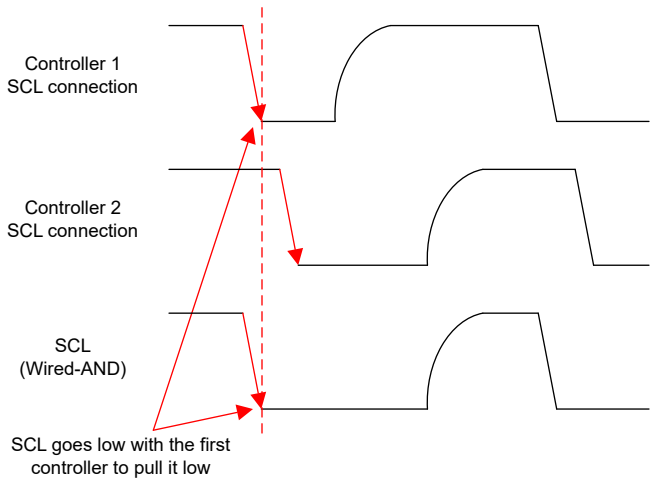



**图 6-2. I²C 时钟同步 SCL 变为低电平**

在 START 条件之后，控制器 1 释放 SCL 以变为高电平。然而，控制器 2 仍然将 SCL 保持为低电平。由于有线 AND 连接，SCL 保持低电平，直到控制器 2 释放 SCL 为高电平。同时，控制器 1 仍在监测 SCL，必须等待另一个控制器释放时钟。在控制器 2 释放 SCL 并变为可用之前，控制器 1 无法推进 SCL 脉冲。

当多个控制器争用总线时，SCL 保持低电平的时间与任何控制器下拉 SCL 的最长时间相同。只有当所有控制器都释放 SCL 后，才能在串行时钟脉冲中将线路释放为高电平。这将同步所有控制器的串行时钟的开始。图 6-3 展示了两个控制器都释放 SCL 时产生的 SCL。

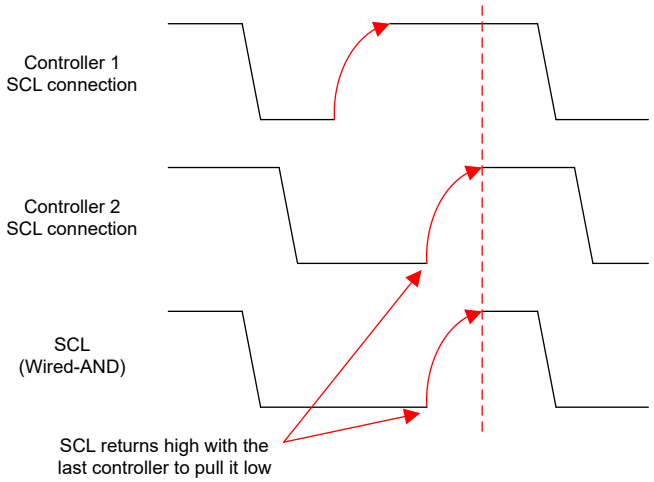



**图 6-3. I²C 时钟同步 SCL 返回高电平**

为了进行时钟同步，每个控制器器件必须监控 SCL 线路，并对 SCL 与预期 SCL 输出不匹配的情况做出反应。


English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

同样，在串行时钟脉冲开始后，所有控制器均下拉 SCL 以完成串行时钟脉冲。同样，通过有线 AND 连接，SCL 随后被下拉，第一个响应控制器通过下拉 SCL 进行响应。完成 SCL 高电平时间周期的第一个控制器决定了有线 AND 连接的 SCL 的高电平时间。图 6-4 显示了 SCL 再次被拉至低电平时的 SCL 线。

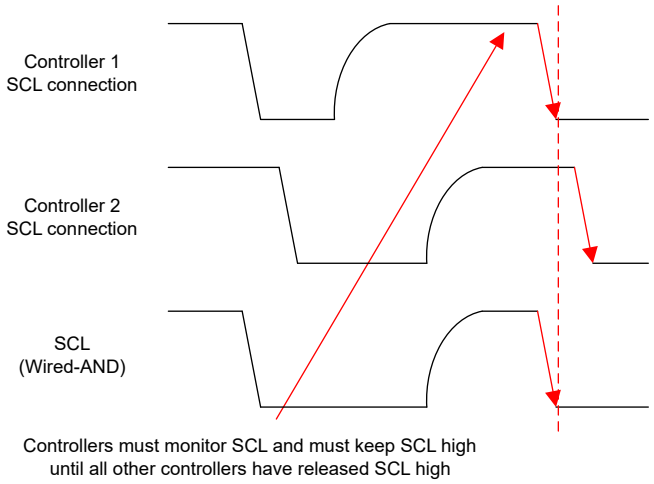



**图 6-4. I²C 时钟同步监控 SCL**

对于所有有效控制器之间的后续时钟脉冲，SCL 时钟的同步会继续。每个 SCL 时钟脉冲的生成周期由具有最长时钟低电平周期的控制器决定，高电平周期由具有最短时钟高电平周期的控制器决定。时钟同步通过图 6-5 中所示的通信继续进行。

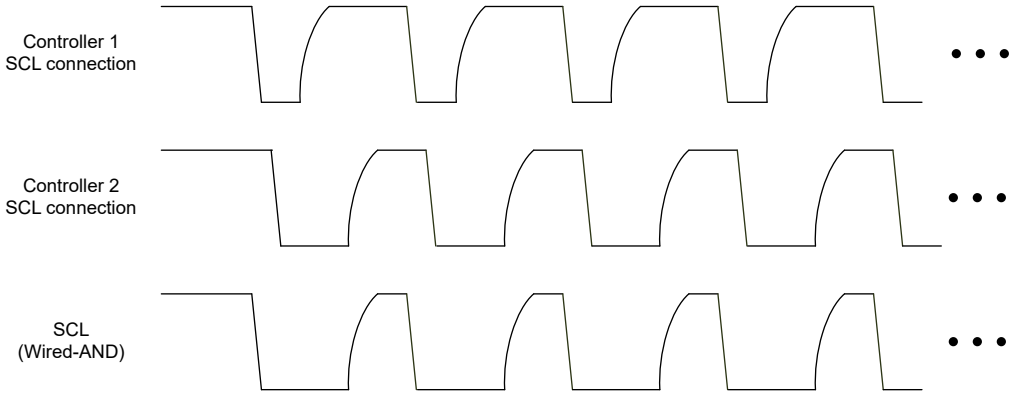



**图 6-5. I²C 时钟同步产生有线 And**

时钟同步的工作原理是，控制器会监视 SCL 线路的每个脉冲，并对 SCL 线路与控制器预期状态不匹配的情况做出反应。



English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

现在串行时钟已同步，仲裁在 SDA 上完成。两个控制器都在 SDA 上正常传输数据，并将其通信发送到预期的目标器件。与 SCL 类似，SDA 是有线 AND 连接。图 6-6 展示了 SDA 在时钟同步后的仲裁。

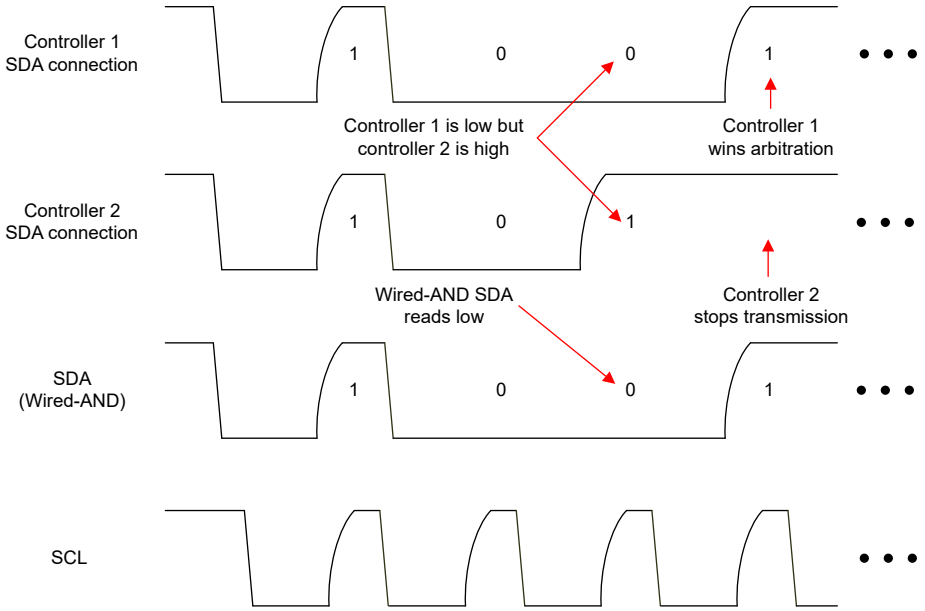



**图 6-6. I²C 控制器仲裁**

在总线仲裁中，通信将持续进行，直到发送的数据之间存在已知差异为止。

在图中，两个器件同时发送 I²C START 条件。对于传输的前两位，数据相同。在数据达到第三位后，存在差异，因为控制器 1 发送 0，而控制器 2 发送 1。由于两个控制器都监控 SDA 和 SCL 线路，因此会发现争用。控制器 2 尽管发送 1，但发现 SDA 为低电平。为了使用正确的有线 AND 结果保持总线上的通信，控制器 2 释放总线，而控制器 1 在仲裁中胜出。


Copyright © 2026 Texas Instruments Incorporated

## 6.2 时钟延展

在一些 I²C 目标器件中，存在目标器件控制 SCL 串行时钟的情况。在这些情况下，目标器件可能会减慢通信速度。此过程称为时钟延展。

通常，SCL 线以及 I²C 时钟速率由控制器控制。但是，在某些情况下，目标器件无法与时钟速率相符。例如，目标器件需要额外的时间来处理命令或发送数据。在这种情况下，目标器件会通过时钟延展来减慢通信速度。

借助时钟延展，在控制器发送一个数据字节后，目标器件会保持 SCL 更长的时间，因此需要控制器来调整时钟。SCL 的这种操作与时钟同步类似。控制器会监测 SCL，如果在控制器释放时钟后 SCL 仍为低电平，则强制延长 SCL 脉冲。则目标器件可以对任何 SCL 脉冲进行时钟延展。然而，时钟延展的一般实现是在发送 ACK 位时通过 SCL 脉冲完成的。

根据 I²C 规范，对目标按住 SCL 进行时钟拉伸没有时间限制。其他类似规范（如 SMBus）对 SCL 保持低电平的时间有限制。

图 6-7 示出了目标器件对 SCL 进行时钟延展的示例。在此示例中，控制器发出 START 并发送目标器件地址。

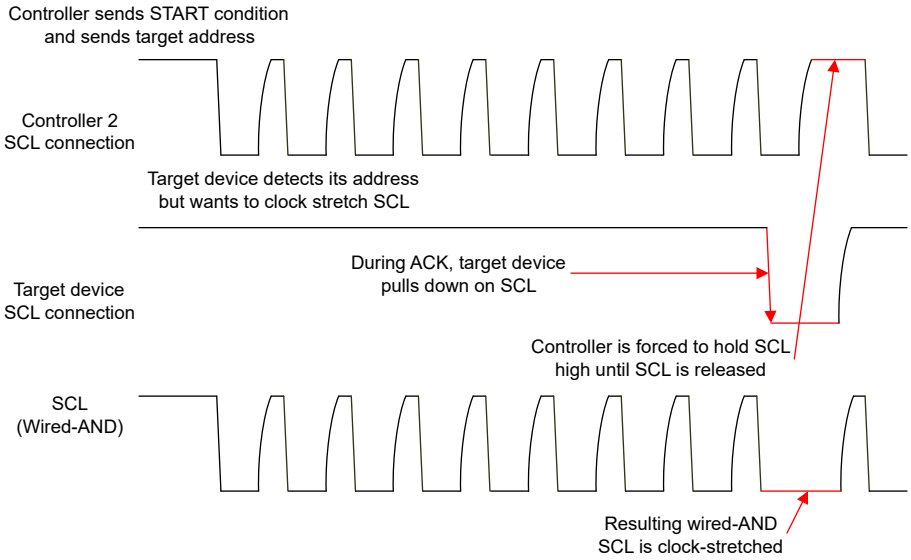



**图 6-7. I²C 目标时钟延展**

当目标器件识别到控制器正在发送正确的目标地址时，目标器件会对该地址进行 ACK。如果需要时钟延展来减慢通信，目标器件可在 ACK 期间下拉 SCL 上的电阻。这是目标器件可以控制 SCL 的唯一实例。

当目标器件开始时钟延展时，即使控制器已释放 SCL，SCL 也保持低电平。由于目标器件可以控制时钟，因此在目标释放 SCL 之前，控制器无法继续处理 SCL 脉冲。控制器继续监测 SCL。一旦 SCL 释放为高电平，控制器就可以继续超过目标器件的 ACK，并继续下一个字节传输。由此产生的 SCL 布线 AND 连接显示 SCL 已延展。数据传输被目标设备延迟，而不会中断通信。



English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

## 6.3 电气规格

每个 I²C 器件的数据表都有电气规格，这些规格涵盖了 I²C 总线的特性。由于 I²C 是通用协议，因此这些规格因器件而异。本节讨论了 I²C 规范中所示的电气特性。

本应用手册并未详细介绍每种规范，而是概述了这些规范的组织结构。I²C 器件的数据表介绍了有关运行 TI 器件所需条件的规格。

例如，表 6-2 显示了 I²C 数字线路的 ADS1119 数字输入和输出的电气特性。

**表 6-2. ADS1119 数字输入和输出的电气特性**

| 参数 | | 测试条件 | 最小值 | 典型值 | 最大值 | 单位 |
|---|---|---|---|---|---|---|
| $V_{IL}$ | 逻辑输入电平，低 | | DGND | | 0.3 DVDD | V |
| $V_{IH}$ | 逻辑输入电平，高 | 2.3V ≤ DVDD < 3.0V，<br>SCL、SDA、A0、A1、$\overline{DRDY}$ | 0.7 DVDD | | DVDD + 0.5 | V |
| | | 3.0V ≤ DVDD ≤ 3.0V，<br>SCL、SDA、A0、A1、$\overline{DRDY}$ | 0.7 DVDD | | 5.5 | V |
| | | $\overline{RESET}$ | 0.7 DVDD | | DVDD | V |
| $V_{hys}$ | 施密特触发输入的迟滞 | 快速模式、超快速模式 | 0.05 DVDD | | | V |
| $V_{OL}$ | 逻辑输出电平，低 | $I_{OL}$ = 3mA | DGND | 0.15 | 0.4 | V |
| $I_{OL}$ | 低电平输出电流 | $V_{OL}$ = 0.4V，标准模式、快速模式 | 3 | | | mA |
| | | $V_{OL}$ = 0.4V，超快速模式 | 20 | | | |
| | | $V_{OL}$ = 0.6V，快速模式 | 6 | | | |
| Ii | 输入电流 | DGND + 0.1V < $V_{Digital\ Input}$ < DVDD – 0.1V | -10 | | 10 | µA |
| Ci | 电容 | 每个引脚 | | | 10 | pF |

该表突出显示了一些参数，给出了 SCL 和 SDA 的低电平和高电平输入和输出电压规格。这规定了每个 I²C 总线都有一个可正确发送和接收高电平和低电平的电压范围。此表还给出了器件开路消耗下拉 SCL 和 SDA 上的最小输出电流。根据所使用的 I²C 模式，还有一些具体特性。

无论使用何种 I²C 器件，这些 SCL 和 SDA 总线特性都可在各自的数据表中找到。数据表提供了足够的这些特性来正确设置器件。有关更多信息，请参阅 I²C 规范。

## 6.4 电压电平转换

设计大型系统的一个常见问题是系统内不同电压电平的混合。例如，当控制器和目标器件在不同的电压下运行时会发生什么情况？

较大的系统可以具有多个存在多个电压的电源。这些不同的电压可为不同的 I²C 控制器及目标器件供电。本节讨论了电压电平转换以及这些不同的 I²C 电压如何相互作用。

电源中的电压不匹配会破坏通信，甚至损坏器件。上拉电阻器的连接决定了一个器件的输出电压是过驱还是欠驱下一个器件的输入。以下示例显示了不匹配的一些后果。


Copyright © 2026 Texas Instruments Incorporated

### 6.4.1 示例 1

在图 6-8 中，控制器和上拉电阻设置为 3.3V，而目标器件设置为 5.0V。

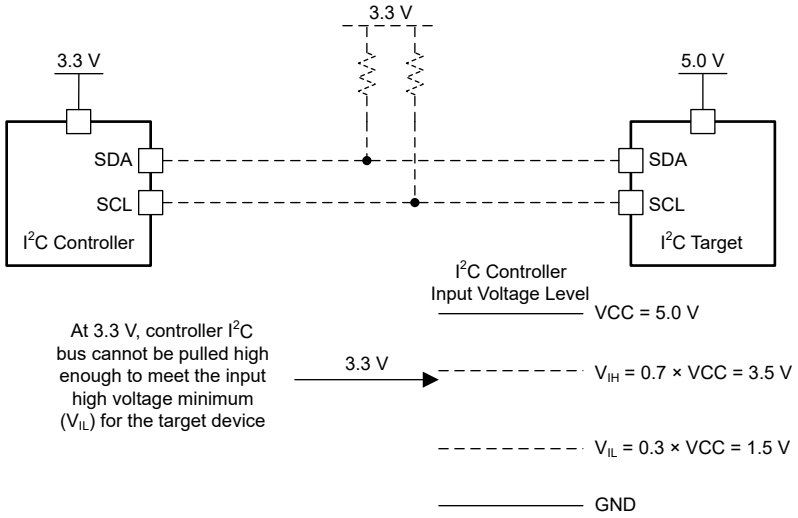



**图 6-8. 总线电压不匹配时的 I²C 上拉电阻器驱动下输入**

在 I²C 规范中，数字输入电压准确解读为数字高电平或低电平需要最小和最大电压。例如，当输入低于最大 0.3 × VCC 时，SDA 和 SCL 被解读为数字输入低电压。同样，当输入高于 0.7 × VCC 的最小值时，SDA 和 SCL 被解读为数字输入高电压。后一种规格对于失配的电源很重要。

当上拉电阻连接到 3.3V 的较低电源时，电阻器无法上拉至高于 3.5V 的最小所需电压。在这种情况下，SDA 和 SCL 都设计为足够高以被作为数字高电平接收。这可能会阻止器件之间的通信。



English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

### 6.4.2 示例 2

在图 6-9 中，控制器设置为 1.8V，但上拉电阻器和目标器件设置为 5.0V。

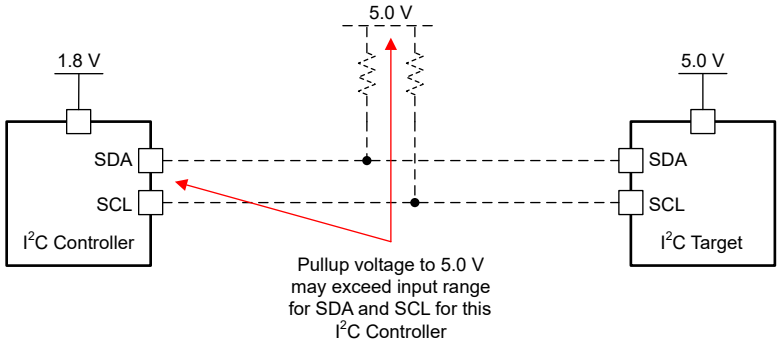



**图 6-9. 总线电压不匹配时的 I²C 上拉电阻器过驱动输入**

在此示例中，I²C 总线可上拉至 5.0V。但是，其中一个器件不可承受如此高的电压。如果器件电压之间的差异过大，电压较低的器件可能会因过压连接而损坏。

### 6.4.3 示例 3

在图 6-10 中，控制器和上拉电阻设置为 5V，但目标器件设置为 3.3V。

I²C 总线线路能够上拉至 5V，超过目标器件电源电压。不过，目标器件的输入可承受更高的电压。这种更高容差是某些 I²C 器件中的一项特性。该功能允许在 I²C 总线之间通过上拉电阻器直接连接至更高电压的电源。请查阅器件数据表，了解此可能的特性。

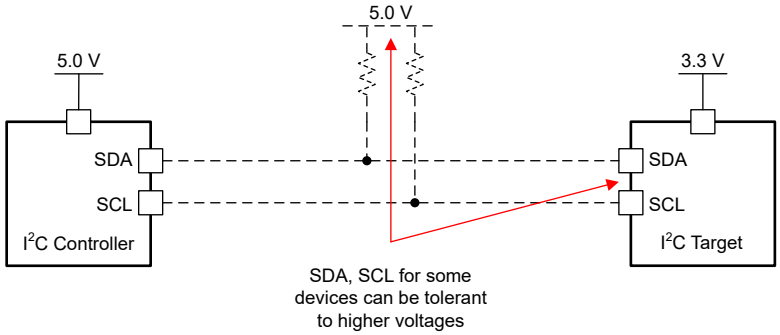



**图 6-10. 在耐受更高电压的线路上对 I²C 上拉电阻器进行过驱动**

例如，ADS1115 只是一个具有 SDA 和 SCL 线的器件，可承受高于电源的电压。查看数据表中的“绝对最大值”表，无论电源电压如何，最大数字输入电压均为 5.5V。借助这种类型的 I²C 线，目标器件可以承受高于电源的上拉电压。这样，即使在不同的电源电压下，器件之间也可进行 I²C 通信。


English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

### 6.4.4 示例 4

如果电源电压不匹配，最好使用特殊器件来桥接两个电源电压。图 6-11 显示了一个使用 I²C 电压电平转换器桥接两个不同电源电压之间的通信的示例。有两组上拉电阻，每个电压电平对应其中一组。作为共模电压转换器，PCA9306 可实现不同电源电平之间的通信。

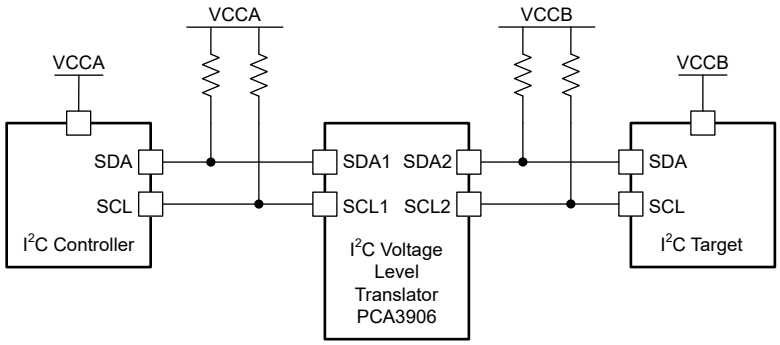



**图 6-11. PCA3906 I²C 电压电平转换器**

## 6.5 确定上拉电阻器大小

若要将系统设计为使总线速度足够快，以满足协议总线速度，请计算上拉电阻的值。

对于 SDA 和 SCL 的开漏连接，这些线路从高电平到低电平以及从低电平到高电平的转换取决于来自器件开漏连接的电流吸收器、总线电容和上拉电阻值。根据这些不同的参数，可计算 I²C 总线速度的最小和最大电阻。

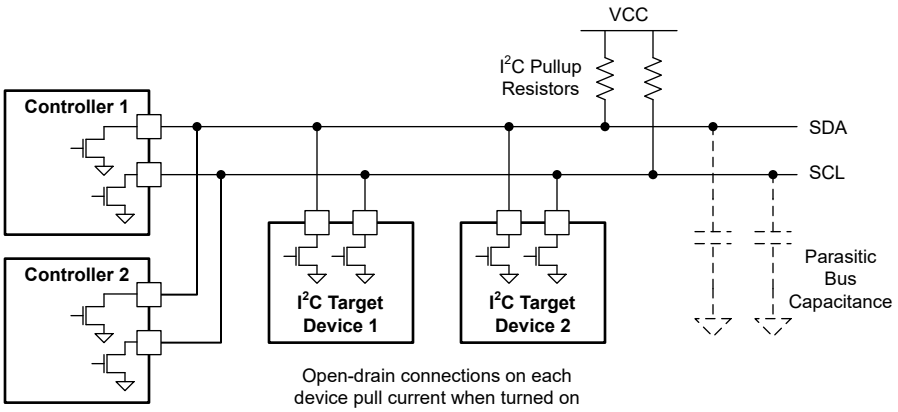



**图 6-12. 影响确定上拉电阻器大小的因素**

正常的上拉电阻器建议为 1kΩ 至 10kΩ。电阻较高时，I²C 通信速度较慢。电阻较低时，I²C 通信需要更多功率。根据几个不同的参数，可计算 I²C 总线速度的最小和最大电阻。



English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

表 6-3 列出了 I²C 总线的一些参数特性。该表列出了 I²C 总线的比特率，总线的最大上升时间和总线上的最大容性负载。所有这些参数都用于确定最小和最大上拉电阻值。

**表 6-3. I²C 协议的参数特性**

| 参数 | | 标准模式（最大值） | 快速模式（最大值） | 超快速模式（最大值） | 单位 |
|---|---|---|---|---|---|
| $f_{SCL}$ | SCLK 时钟频率 | 0 至 100 | 0 至 400 | 0 至 1000 | kHz |
| $t_r$ | SDA 和 SCL 信号的上升时间 | 1000 | 300 | 120 | ns |
| $C_b$ | 每个总线的容性负载 | 400 | 400 | 550 | pF |

除了这些参数外，还考虑了 I2C 输入和输出电压最小值和最大值。表 6-4 介绍了这些电压。

**表 6-4. SDA 和 SCL 输入和输出电压的特性**

| 参数 | | 标准模式 | | 快速模式 | | 超快速模式 | | 单位 |
|---|---|---|---|---|---|---|---|---|
| | | 最小值 | 最大值 | 最小值 | 最大值 | 最小值 | 最大值 | |
| $V_{IL}$ | 低电平输入电压 | -0.5 | 0.3 × VCC | -0.5 | 0.3 × VCC | -0.5 | 0.3 × VCC | V |
| $V_{IH}$ | 高电平输入电压 | 0.7 × VCC | VCC + 0.5 | 0.7 × VCC | VCC + 0.5 | 0.7 × VCC | VCC + 0.5 | V |
| $V_{OL}$ | 低电平输出电压，3mA 吸收电流，VCC > 2V | 0 | 0.4 | 0 | 0.4 | 0 | 0.4 | V |
| | 低电平输出电压，3mA 灌电流，VCC ≤ 2V | - | - | 0 | 0.2 × VCC | 0 | 0.2 × VCC | V |

### *6.5.1 最小上拉电阻大小*

图 6-13 展示了 I²C 总线的开漏连接以及 SDA 或 SCL 的输出波形。SDA 和 SCL 总线从器件的电流拉低。

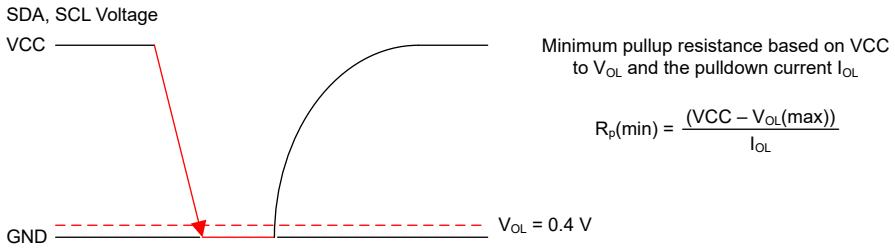



**图 6-13. 基于下拉电流的最小上拉电阻**

当器件释放 SDA 或 SCL 时，总线连接到 $V_{CC}$ 电压。当有源时，器件漏极将拉取总线线路输出至近地端。输出必须降至输出低电平电压，即 $V_{OL}$。该器件通过电流 $I_{OL}$ 将总线线路拉至低电平。表 6-4 中介绍了 $V_{OL}$ 和 $I_{OL}$（3mA 灌电流）。根据此电流，计算上拉电阻所需的最小电阻。如果电阻较小，则输出电流无法将总线的输出电压拉低到足以被识别为数字低电平。如方程式 4 中所示。

$$R_P(min) = (VCC - V_{OL}(max)) / I_{OL} \quad (4)$$

求解最小上拉电阻，从 3.3V 电源电压中减去 0.4V 的输出低电平电压。然后除以 3mA 总线拉取的电流。这样可得到 967 Ω 为最小电阻。

### *6.5.2 确定最大上拉电阻器大小*

开漏连接释放输出电流后，上拉电阻器将总线连接拉至高电平。总线线路输出波形呈指数稳定。当电阻器从地拉高电压时，电压稳定时间基于总线电容 ($C_B$)。由于 I²C 标准上升时间规格，最大上拉电阻受总线电容的限制。在电阻较高的情况下，上拉输出上升得太慢，无法足够快地达到逻辑高电平。


English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

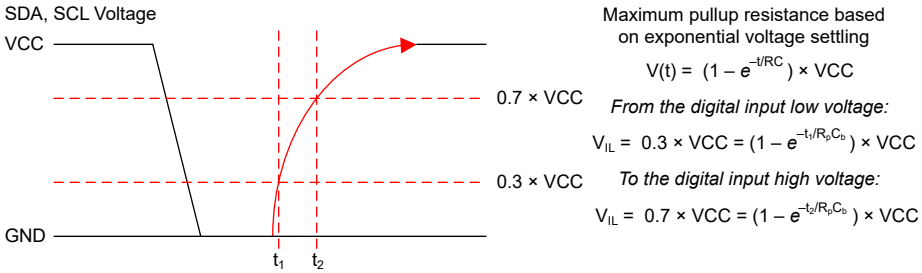



**图 6-14. 基于上拉电阻上升和总线电容的最大上拉电阻器**

随时间呈指数稳定的公式如方程式 5 所示，其中包含上拉电阻。

$$V(t) = \left(1 - e^{-t/RC}\right) \times VCC \tag{5}$$

上升时间基于从电源电压 0.3 倍的数字输入低电压 ($V_{IL}$) 到电源电压 0.7 倍的数字输入高电压 ($V_{IH}$) 的转换。表 6-3 中描述了上升时间，而表 6-4 中描述了 $V_{IL}$ 和 $V_{IH}$。使用这些参数实现上拉稳定会得到方程式 6 和方程式 7。

$$V_{IL} = 0.3 \times VCC = \left(1 - e^{-t_1/R_P C_B}\right) \times VCC \tag{6}$$

$$V_{IL} = 0.7 \times VCC = \left(1 - e^{-t_2/R_P C_B}\right) \times VCC \tag{7}$$

根据指数建立公式，可以根据最大上拉电阻和总线电容来求解上升时间。在本例中，计算方法是针对 400 pF 总线电容（针对最大总线电容）和 3.3V 电源电压。方程式 8 和方程式 9 求解上升时间，然后求解最大上拉电阻器。

$$t_{RISE} = t_2 - t_1 = 0.8473 \times R_P \times C_B \tag{8}$$

$$R_P(max) = t_{RISE} / (0.8473 \times C_B) \tag{9}$$

上升时间取决于 I²C 模式。在此示例中，可以使用标准模式。取上升时间 1000 纳秒除以 0.8473 乘以 400pF 的数量。这样，可得出最大电阻为 2.95kΩ。

在最小电阻为 967Ω 且最大电阻为 2.95kΩ 的情况下，这些值看起来提供了一个较窄的电阻范围。但是，这个较小范围是因为计算上拉电阻大小时，可在 400pF 的最大标准模式总线电容下运行。该总线电容值异常大，尤其是对于电路板上的寄生电容而言。如果设计中的总线电容较低（有可能），则可以增大最大电阻，从而降低 I²C 总线上的功率耗散。

有关 I²C 上拉电阻计算的更详细说明，请参阅 *I2C 总线上拉电阻计算* 应用报告。*模拟工程师计算器* 中也提供了 I²C 上拉计算器。



English Document: SBAA565
Copyright © 2026 Texas Instruments Incorporated

# 7 与 I²C 类似的协议

I²C 规范讨论了其他几种基于 I²C 的通信协议。这些其他协议可以与 I²C 通信相似并兼容，可用于特定应用。一些协议还为其系统定义了命令集和应用特定的扩展。本部分简要介绍了这些其他协议的应用，但其系统、应用程序和用途可在其他地方找到。

第一种类似的协议是两线制接口，即 TWI。该协议与 I2C 基本相同。但是，由于 TWI 不支持 START 字节并且不支持高速模式这两个方面，存在一些细微差异。通常，与 TWI 兼容的器件应该与 I²C 兼容，并且可以使用相同的逻辑分析仪看到该协议。

系统管理总线 (SMBus®) 是类似于 I²C 的协议，专为特定功能定制。SMBus 通常用于服务器和计算机主板以进行电源管理。该协议在通信协议中与 I²C 非常相似，并且可以通过 I²C 控制器理解。

与 I²C 相比，SMBus 协议具有一些附加功能。SMBus 可以动态设置地址，从而在系统启动时实现快速通信。此外，总线有一个 35ms 的超时时间，可防止一个器件无限期连接总线。该协议还具有数据包错误检查功能，可在数据通信中进行错误检测。有一条名为 SMBAlert 的额外线路，目标器件使用该线路作为中断，向控制器告知目标器件检测到的某些事件。

电源管理总线 (PMBus)® 是由 Intel 定义的 SMBus 的变体。PMBus 用于电源的数字管理。此协议还定义了用于检索系统中电压、电流和功率相关数据的特定命令。

智能平台管理接口 (IPMI) 是另一种基于 I²C 的协议。基板管理控制器 (BMC) 使用此协议来监控和管理系统 CPU、固件及操作系统的自主计算机子系统。该协议对计算机主板或者服务器使用基于消息的标准化接口。即使在主系统关闭时，BMC 也一直在运行。这可实现系统的操作、测量和远程管理。

I²C 规范中还讨论了其他几种类似协议。高级电信计算架构 (ATCA) 是 Compact PCI 的后续组件，用于机架式电信硬件。显示数据通道 (DDC) 是主机用于控制显示功能的监视器或者显示信息协议。最后，C-Bus 是另一个从 I²C 衍生出的协议。如保留地址部分所述，该协议在全球的某些地区用于家居和楼宇自动化，但通常不在美国使用。

# 8 总结

I²C 是各种产品中所使用的通用数字通信标准。该协议使用两线制通信接口，允许多个控制器和多个目标外设。本应用手册介绍了该协议的许多重要方面，作为使用 I²C 与控制器器件通信的指南。

本应用手册讨论了 I²C 通信的协议和物理层。I²C 通常与数据转换器器件一起使用，因此为 DAC 和 ADC 提供了寄存器的写入和读取示例。本应用手册还介绍了协议的许多不太常见的方面，例如器件中通常未实现的时钟延展、快速模式和时钟延展。

此处介绍的主题并未全部深入介绍。不过，本应用手册为系统设计人员提供了该协议的实际应用知识。使用这些信息，设计人员可以设置其 I²C 系统，并在出现通信问题时对其进行调试。


Copyright © 2026 Texas Instruments Incorporated

# 重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

版权所有 © 2026，德州仪器 (TI) 公司

最后更新日期：2025 年 10 月